Δοκιμή και Αξιοπιστία Ηλεκτρονικών Κυκλωμάτων

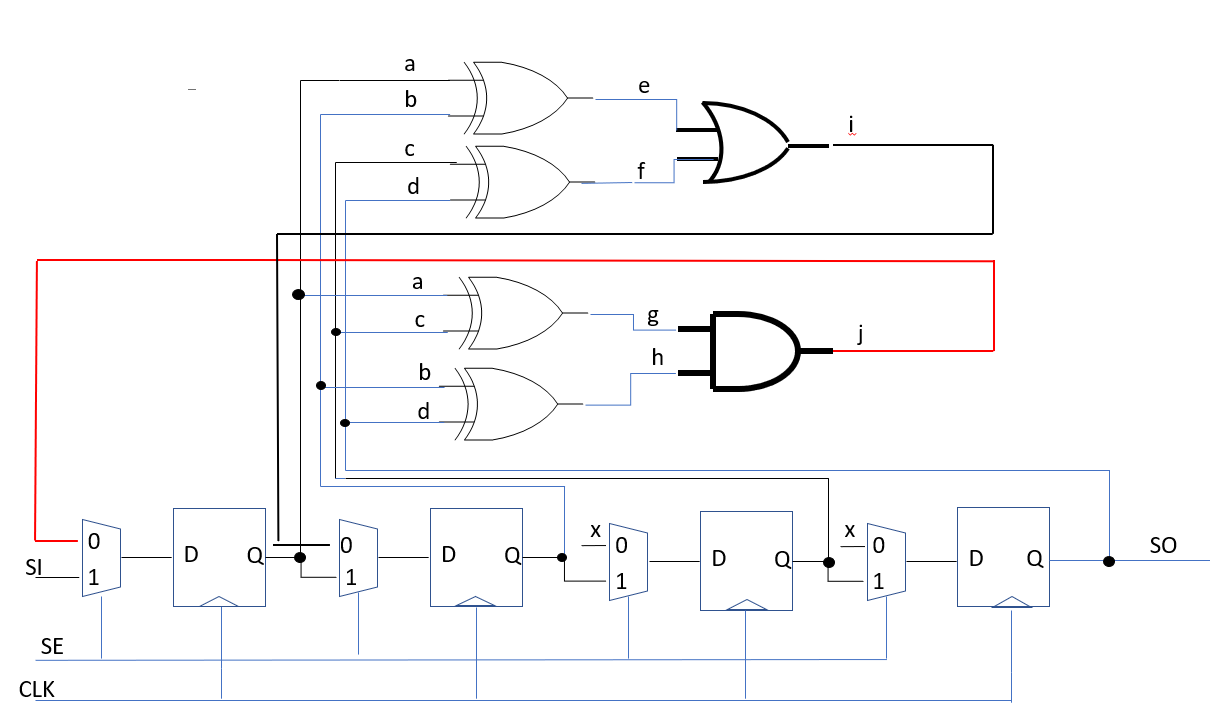
Άσκηση 1

Τζιάστα Θεοδώρα

ΑΜ: 4178

**Άσκηση 1.1**

Για αυτό το ερώτημα της άσκησης σχεδιάστηκε το παρακάτω TRCUT:



Στο παραπάνω σχήμα φαίνεται η scan chain αποτελούμενη από 4 scan D Flip Flop, οι έξοδοι των οποίων αποτελούν τις εισόδους του CUT. Πιο αναλυτικά υπάρχουν 4 SDFF:

1o: Raj: Η έξοδος αυτού του SDFF είναι η είσοδος a του CUT και χρησιμοποιείται ως controllability point για να την ελέγχει. Επιπλέον αυτό το SDFF χρησιμοποιείται και ως observability point της εξόδου j του CUT.

2o: Rbi: Η έξοδος αυτού του SDFF είναι η είσοδος b του CUT και χρησιμοποιείται ως controllability point για να την ελέγχει. Επιπλέον αυτό το SDFF χρησιμοποιείται και ως observability point της εξόδου i του CUT.

3o: Rc: Η έξοδος αυτού του SDFF είναι η είσοδος c του CUT και χρησιμοποιείται ως controllability point για να την ελέγχει. Αυτό το scan cell δεν χρησιμοποιείται ως observability point.

4o: Rd: Η έξοδος αυτού του SDFF είναι η είσοδος d του CUT και χρησιμοποιείται ως controllability point για να την ελέγχει. Αυτό το scan cell δεν χρησιμοποιείται ως observability point. Η έξοδος του αποτελεί και την SO όλου του TRCUT.

Κάθε ένα από τα scan cells λειτουργεί σε δύο modes. Όταν το SE=1 τότε γίνεται η σειριακή σάρωση κατά την οποία ολισθαίνουν νέα δεδομένα από την είσοδο SI, ενώ όταν το SE=0 γίνεται η κανονική λειτουργία στην οποία φορτώνονται οι αποκρίσεις του συνδυαστικού κυκλώματος στην scan chain.

Προκείμενου να υλοποιηθεί το παραπάνω κύκλωμα αρχικά δημιουργήθηκε σε Verilog η μονάδα της συνδυαστικής λογικής με τον παρακάτω κώδικα:

module comb\_logic(a,b,c,d,i,j);

input a,b,c,d;

output i,j;

wire e,f,g,h;

xor(e,a,b);

xor(f,c,d);

xor(g,a,c);

xor(h,b,d);

**and**(i,e,f);

**or**(j,g,h);

endmodule

Στη συνέχεια υλοποιήθηκε ολόκληρο το TRCUT και φαίνεται στον παρακάτω κώδικα:

module trcut(CLK,SE,SI,SO);

input CLK,SE,SI;

output SO;

wire a\_cp,b\_cp,c\_cp,d\_cp; //connection comb\_logic **with** scan chain

assign SO=d\_cp;

comb\_logic CLinstance(a\_cp,b\_cp,c\_cp,d\_cp,i,j);

//scan chain

SDFF R\_AJ(CLK, j, SI, SE, a\_cp);

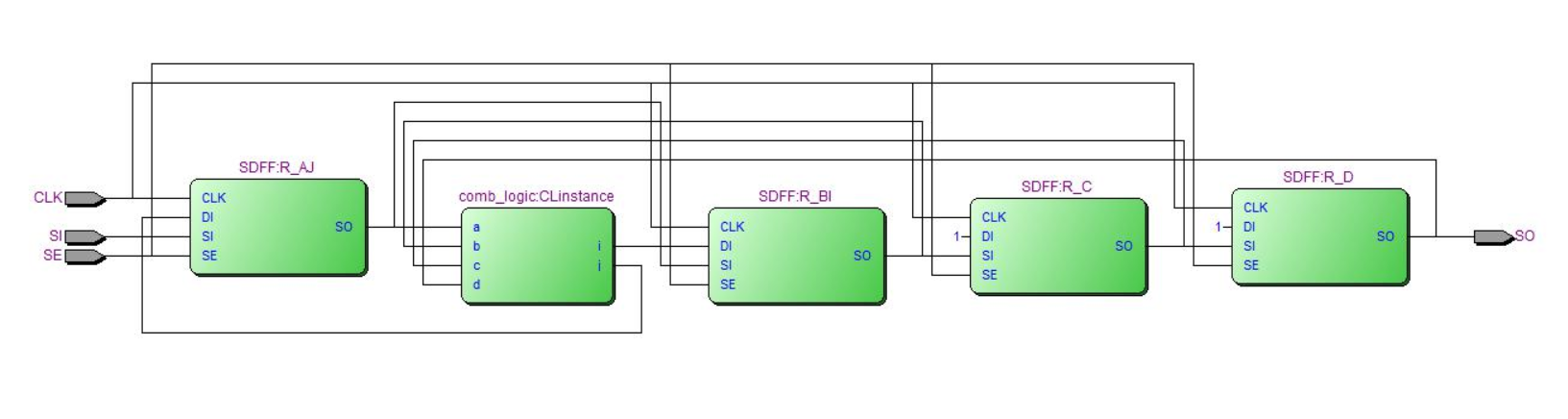
SDFF R\_BI(CLK, i, a\_cp, SE, b\_cp);

SDFF R\_C(CLK, x\_value, b\_cp, SE, c\_cp);

SDFF R\_D(CLK, x\_value, c\_cp, SE, d\_cp);

endmodule

Έτσι το σχηματικό που προκύπτει από το Quartus είναι το παρακάτω:



Έπειτα δημιουργήθηκε testbench που ακολουθεί για να επιβεβαιώσει την ορθή λειτουργία του κυκλώματος:

`timescale **1**ns/**1**ps

module trcut\_tb();

reg CLK,SEtb,data\_in;

wire SO;

integer counter;

reg [**3**:**0**]SItb;

reg [**3**:**0**]sec\_si;

trcut trcut\_instance(CLK,SEtb,data\_in,SO);

//Block **for** clock generation

initial begin

CLK=**0**;

#20

forever begin

#10 CLK=!CLK;

end

end

initial begin

SItb=**4**'b0101; // 1 is the LSB

sec\_si=**4**'b1001;

counter=**0**;

end

initial begin

//shift SItb=**4**'b0101

repeat(**4**) @(posedge CLK)

**if**(counter<**4**) begin

SEtb=**1**; //scan mode

data\_in<=SItb[counter]; //shift each bit of the test vector

counter<=counter+**1**;

end

repeat (**1**)@(posedge CLK)

SEtb=**0**; //capture mode

//shift sec\_si=**4**'b1001

//SO take the captured data

counter=**0**;

repeat(**4**) @(posedge CLK)

**if**(counter<**4**) begin

SEtb=**1**;//scan mode

data\_in<=sec\_si[counter];//shift **1001**

counter<=counter+**1**;

end

repeat (**1**)@(posedge CLK)

SEtb=**0**; //capture mode

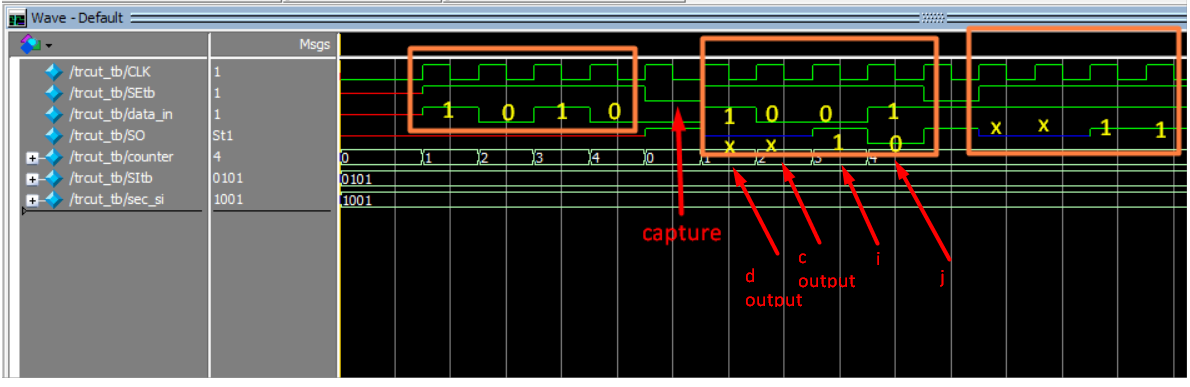
repeat (**1**)@(posedge CLK)

SEtb=**1**;//scan mode

end

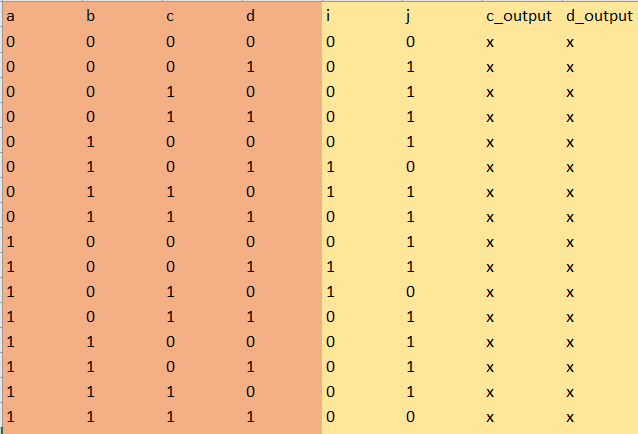
endmodule

Στο παραπάνω testbench έχοντας ενεργοποιημένο το scan mode με SEtb=1 για 4 κύκλους ρολογιού εισάγεται σειριακά μέσω της εισόδου data\_in το διάνυσμα SItb=0101 με LSB το δεξιότερο bit, δηλαδή το 1. Αφού φορτωθεί το διάνυσμα ενεργοποιείται το capture mode βάζοντας SEtb=0 για 1 κύκλο ρολογιού φορτώνοντας τις αποκρίσεις της συνδυαστικής λογικής στη scan chain. Έπειτα ενεργοποιείται και πάλι το scan mode για ακόμα 4 παλμούς ρολογιού, ολισθένεται το νέο διάνυσμα sec\_si=1001 με LSB το δεξιότερο bit καθώς στην έξοδο SO λαμβάνονται οι αποκρίσεις που φορτώθηκαν στη scan chain. Σύμφωνα με το συνδυαστικό λογικό κύκλωμα έχοντας στις εισόδους του το διάνυσμα 0101 περιμένουμε στην έξοδο SO 01xx ενώ εισάγοντας το 1001 η έξοδος πρέπει να εμφανίσει 11xx. H έξοδος SO θα λάβει στον κύκλο αμέσως μετά το capture την έξοδο του scan cell R\_D έπειτα του R\_C, ύστερα του R\_BI και τέλος του R\_AJ. Τα δύο τελευταία scan cells δεν χρησιμοποιούνται ως observation points επομένως οι τιμές τους θα είναι αδιάφορες.Η διαδικασία επαναλαμβάνεται για όσα test vectors εισάγουμε. Όλα τα παραπάνω γίνονται αντιληπτά στις κυματομορφές που ακολουθούν:



**Άσκηση 1.2**

Για την κατασκευή του δεύτερου ερωτήματος είναι απαραίτητη η δημιουργία του πίνακα αληθείας του CUT ο οποίος φαίνεται παρακάτω:



Έπειτα υλοποιήθηκε το testbench το οποίο εφαρμόζει στην είσοδο του CUT κάθε διάνυσμα του πίνακα αληθείας και λαμβάνει τις αποκρίσεις του. Φαίνεται παρακάτω:

`timescale **1**ns/**1**ps

module trcut\_truthtable\_tb();

reg CLK,SEtb,data\_in,confirmed\_output;

wire SO;

integer counter,i,j,output\_counter;

reg[**7**:**0**] truth\_table [**0**:**15**];

trcut trcut\_instance(CLK,SEtb,data\_in,SO);

//Block **for** clock generation

initial begin

CLK=**0**;

#20

forever begin

#10 CLK=!CLK;

end

end

initial begin

// out **in**

// | |

// \/ \/

truth\_table[**0**]=**8**'b00xx0000;

truth\_table[**1**]=**8**'b10xx0001;

truth\_table[**2**]=**8**'b10xx0010;

truth\_table[**3**]=**8**'b10xx0011;

truth\_table[**4**]=**8**'b10xx0100;

truth\_table[**5**]=**8**'b01xx0101;

truth\_table[**6**]=**8**'b11xx0110;

truth\_table[**7**]=**8**'b10xx0111;

truth\_table[**8**]=**8**'b10xx1000;

truth\_table[**9**]=**8**'b11xx1001;

truth\_table[**10**]=**8**'b01xx1010;

truth\_table[**11**]=**8**'b10xx1011;

truth\_table[**12**]=**8**'b10xx1100;

truth\_table[**13**]=**8**'b10xx1101;

truth\_table[**14**]=**8**'b10xx1110;

truth\_table[**15**]=**8**'b00xx1111;

counter=**0**;

output\_counter=**4**;

i=**0**;

j=**0**;

SEtb=**1**;

end

initial begin

**for**(i=**0**;i<**16**;i=i+**1**)begin

repeat(**4**) @(posedge CLK)

**if**(counter<**4**)begin

SEtb=**1**;//scan mode

data\_in<=truth\_table[i][counter];//shift each bit of the truth table's vactor

counter<=counter+**1**;

end

repeat (**1**)@(posedge CLK)

SEtb=**0**;//capture mode

//this code made to confirme the output data but

//adds **4** clock cycles at every interation

// repeat (**1**)@(posedge CLK)

// SEtb=**1**;

//

// repeat (**4**)@(posedge CLK)

// **if**(output\_counter<**8**)begin

//

// **if**(SO===truth\_table[i][output\_counter])begin

// confirmed\_output<=**1**;

// end

// **else** begin

// confirmed\_output<=**0**;

// end

// output\_counter<=output\_coun

ter+**1**;

// end

counter=**0**;

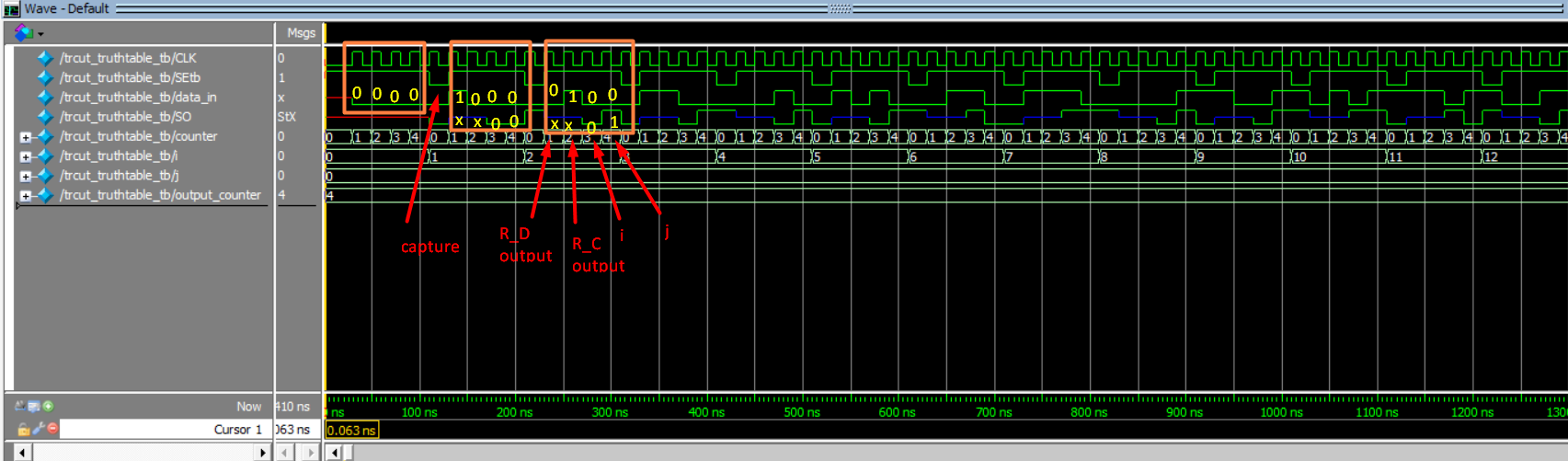
output\_counter=**4**;

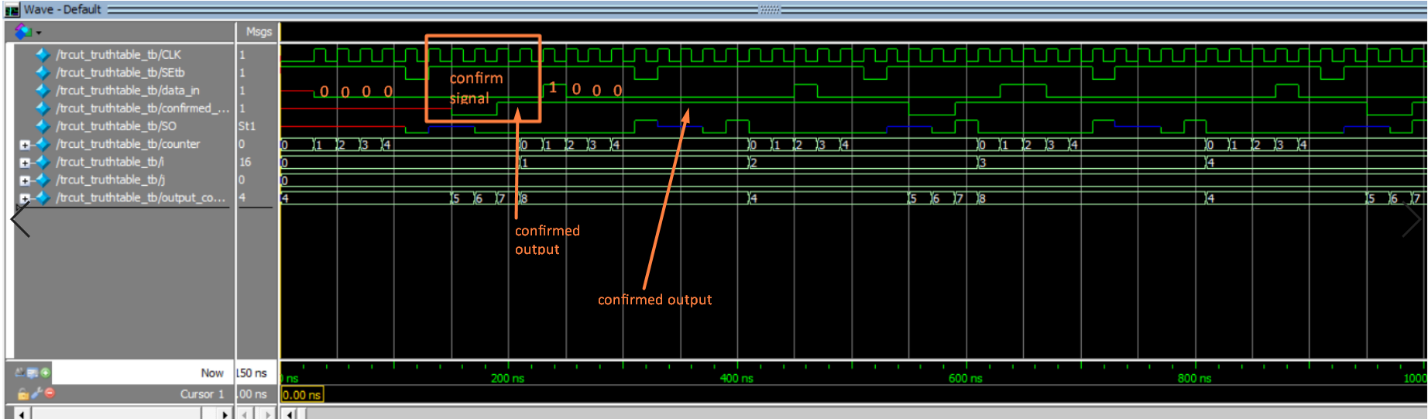
end

end

endmodule

Στον πίνακα truth\_table του παραπάνω testbench κοιτώντας από δεξιά προς τα αριστερά στο slice 3:0 φαίνονται οι είσοδοι abcd ενώ στο slice 7:4 οι έξοδοι των scan cells με το MSB το j και LSB το d. Επιπλέον θα έπρεπε κάθε φορά να επιβεβαιώνεται ότι η έξοδος που λήφθηκε στο SO ότι είναι και η σωστή σύμφωνα με τον πίνακα αληθείας, αυτό δεν μπόρεσα να το υλοποιήσω εντελώς σωστά και ο κώδικας φαίνεται με σχόλια στο παραπάνω testbench. Αυτό που συμβαίνει είναι ότι δεν γίνεται παράλληλα η επιβεβαίωση και η ολίσθηση του επόμενου διανύσματος αλλά αφού γίνει η επιβεβαίωση ολισθαίνεται και το επόμενο διάνυσμα. Παρακάτω φαίνονται δύο εικόνες των κυματομορφών, στην πρώτη χωρίς την υλοποίηση του σήματος επιβεβαίωσης ενώ στην δεύτερη με αυτό. Στην πρώτη φωτογραφία κυματομορφών εισάγεται το κάθε διάνυσμα στην scan chain και αφού γίνει capture οι έξοδοι του CUT που είναι αποθηκευμένες στα scan cells περνούν στην SO με την σειρά που φαίνεται στην φωτογραφία και ταυτόχρονα ολισθαίνεται το επόμενο διάνυσμα(η διαδικασία έχει περιγραφεί και στο προηγούμενο ερώτημα). Η διαδικασία αυτή επαναλαμβάνεται για όλα τα διανύσματα του πίνακα αληθείας. Στην δεύτερη εικόνα παρατηρούνται οι 4 κύκλοι που μεσολαβούν μεταξύ του προηγούμενου και του επόμενου διανύσματος καθώς και το σήμα επιβεβαίωσης το οποίο γίνεται 1 όταν η έξοδος είναι η αναμενόμενη. Όταν οι έξοδοι των δύο τελευταίων scan cells είναι σε τιμές υψηλής εμπέδησης το σήμα επιβεβαίωσης κάποιες φορές ταυτίζεται με αυτές και άλλες όχι και αυτό είναι λογικό.





**Άσκηση 1.3**

Ο χρόνος που απαιτείται για τον έλεγχο ορθής λειτουργίας μιας συσκευής είναι ο χρόνος που απατείται για να ολισθήσουμε το διάνυσμα μεσά στην αλυσίδα και 1 κύκλος ακόμα του capture και αυτό πολλαπλασιασμένο με το πλήθος των διανυσμάτων του πίνακα αληθείας και την περίοδο του ρολογιού. Επομένως θα έχουμε:

* Για 4 εισόδους: χρόνος=(4+1)\*16\*1/f=5\*16\*10\*10-6=800\*10-6=800 ms
* Για 10 εισόδους: χρόνος=(10+1)\*210\*1/f=11264\*10\*10-6=112640 ms
* Για 20 εισόδους: χρόνος=(20+1)\*220\*1/f=22020096\*10\*10-6=220,20096 s
* Για 30 εισόδους: χρόνος=(30+1)\*230\*1/f= 33285996544\*10\*10-6=332857,41824 s
* Για 40 εισόδους: χρόνος=(40+1)\*240\*1/f= 4.5079977e+13\*10\*10-6ms